Big O Notation

Big O notation is a notation used when talking about growth rates. It formalizes the notion that two functions "grow at t
rate," or one function "grows faster than the other," and such.

It is very commonly used in computer science, when analyzing algorithms. Algorithms have a specific running time, usua
declared as a function on its input size. However, implementations of a certain algorithm in different languages may yiel
different function. For example, an algorithm with input size n bytes, when implemented in C++, might take time n^2
microseconds; but when implemented in Python, it might take time $1000n^2 + 1000n$ microseconds due to it being a sl
language. Thus, instead, we often talk about an algorithm's **growth rate**; whether the algorithm only takes time proporti
its input size (linear), or the square of the size (quadratic), or perhaps it doesn't depend on its input size (constant). This i
formalized in big O notation, stating that the algorithm above runs in "quadratic time," or "$O(n^2)$."

Another common use is when talking about approximations in power series. A power series has many terms, usually infi
for the sake of approximations we often don't need too many terms. For example, the sine function has power series $\sin$
$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \cdots$; to approximate $\sin 0.1$ to four digits, we most likely don't need terms with exponent greate
because $0.1^5 = 0.00001$ doesn't contribute to the fourth digit. We formalize this also with big O notation; we state "$\sin$
$x - \frac{x^3}{6} + O(x^5)$," indicating that the terms afterwards are too insignificant to matter.

Intuitive Meaning

Informally, if we have functions f, g such that f eventually grows slower than some multiple of g, we say "$f = O(g)$."

For example, consider the functions $f(n) = 1000n^2$ and $g(n) = n^3$ as $n \to \infty$. "Eventually," namely when $n > 1000$,
that $f(n) < g(n)$, and thus $f(n)$ grows slower than $g(n)$. This means $f = O(g)$, or $1000n^2 = O(n^3)$.

In fact, since it says "some multiple of g," we also have $1000n^2 = O(n^2)$; for some multiple of n^2 $\left(\text{namely } 1001n^2\right)$, ev
(in this case $n > 1$) we have $f(n) < g(n)$.

The above examples talk when the input approaches ∞, as often used in analysis of algorithms. (Algorithms can be usec
arbitrarily large input size, and hence we talk about approaching infinity.) In approximations of power series, we often ta
functions as the input approaches a specific, finite value: for example, approaching 0. We can also use big O notation in
case.

For example, consider the functions $f(n) = 1000n^2$ and $g(n) = n$ as $n \to 0$. "Eventually," which in this case means 0
0.001, we have $f(n) < g(n)$, and thus $f = O(g)$ as $n \to 0$.

Formal Definition

We formalize the informal definition above, mostly by clarifying what "eventually" and "slower" means.

DEFINITION

If $f(x), g(x)$ are functions on the real numbers, then we say $f = O(g)$ or "f is big-oh of g" as $x \to \infty$ if there exist
constants M, c such that $|f(x)| \le c|g(x)|$ for all $x > M$.

If $f(x), g(x)$ are functions on the real numbers, then we say $f = O(g)$ or "f is big-oh of g" as $x \to a$ if there exist s
constants δ, c such that $|f(x)| \le c|g(x)|$ for all $0 < |x - a| < \delta$.